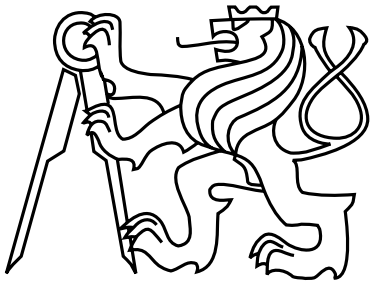
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Diplomová práce

Klasifikace pohybu na základě nasnímaných pohybových dat

Bc. Václav Strnad

Vedoucí práce: Berka Roman Ing., Ph.D.

Studijní program: Počítačová grafika a interakce, Magisterský

Obor: Otevřená informatika - Nový

11. května 2015

Poděkování

Děkuji vedoucímu práce Romanu Berkovi Ing., Ph.D. za vedení diplomové práce a rodině za pomoc a trpělivost při práci na dokumentaci.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 11. 5. 2015 .....

Abstract

Study the movement known classification methodology and design your own way of identifying the selected range of motion in the sequence data input. Inspire design by article, which is include in task. Verify implementation of the application with scanned data on input and the output will indicate motion class of movement. At the end, specify the conditions under which the application is functional. Define at least three classes of motion (walk, run, jump), and perform adequate testing of the data, which will include at least 5 motion sequences for each class. Implement in C ++ environment, perform testing on data from public databases.

Abstrakt

Prostudujte známé metodiky klasifikace pohybu a navrhněte vlastní způsob identifikace vybrané škály pohybů v sekvenci dat na vstupu. Při návrhu se inspirujte článkem, který je součástí zadání. Vlastní návrh ověřte implementací aplikace, která bude na vstupu přijímat nasnímaná data a na výstupu bude indikovat příslušnost pohybu k některé třídě pohybu. Na konci práce specifikujte podmínky, za nichž je aplikace funkční. Pro klasifikátor definujte alespoň tři třídy pohybu (chůze, běh , skok) a testování proveďte na dostatečné množině dat, která bude obsahovat alespoň 5 pohybových sekvencí na každou třídu. Implementujte v prostředí jazyka C++, testování proveďte na datech z volně přístupných databází.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Tato diplomová práce se zabývá problematikou automatické kategorizace pohybu. Jejím úkolem je vytvořit program, který se bude umět naučit rozpoznávat jednotlivé pohyby z dat, které uživatel programu poskytne. Program po fázi učení musí být schopen automaticky rozpoznat, z jakých pohybů se rozpoznávaný záznam skládá.

Téma automatické kategorizace pohybu jsem si vybral ke zpracování proto, že se jedná o velice zajímavý problém, který ještě není dosud plně popsán, a z části i proto, že je to pro mne výzva vytvořit nový program schopný samostatné práce.

Původní inspirací pro vývoj programu mi byl článek zadaný vedoucím práce. Jelikož problematika algoritmu automatické kategorizace pohybu v článku není dostatečně detailně popsána, využil jsem jako další zdroj algoritmy na rozpoznávání obrazu i své znalosti podobých problematik.

Výsledek práce by měl být úvodem do problematiky automatické kategorizace pohybu a jedním z prvků systému pro animátorská studia, která pracují s velkými databázemi nasnímaných pohybů. Tato data mají dosud uložena v jednoduchých strukturách bez podrobnějších popisů, které by jim umožňovaly efektivnější práci s těmito databázemi.

Moje práce se nejprve zabývá problematikou uložených dat pohybu v různých formátech a nasnímané různými technologiemi. Dále řeší problematiku různého způsobu snímání pohybu a rozdílných vlastností dat. V další části práce popisuje implementaci algoritmu a jeho testování.

Výsledný program jsem testoval na datech získaných z veřejné databáze.

Kapitola 2

Popis problému, specifikace cíle

2.1 Naměřená data

„Motion capture“, zkráceně MOCAP, snímá pohyb pomocí bodů umístěných na pohybujícím se jedinci. Nejčastěji se jedná o člověka. MOCAP se využívá hlavně ve filmovém odvětví, případně v lékařství. Snímání polohy jedince je realizováno různými technologiemi, které jsou popsány v následující sekci.

2.1.1 Typy MOCAP

Elektromagnetické

- Poloha bodu je vypočtena na základě polohy tří navzájem na sebe kolmých cívek v magnetickém poli.
- Nevýhodou jsou poruchy magnetického pole způsobené kovovými objekty na scéně.

Optické

- Poloha bodů je vypočtena z projekce do kamer, které natáčí scénu.
- Nevýhodou je nutnost snímat každý bod alespoň třemi kamerami. Objekty na scéně nebo samotný snímaný jedinec tyto body zastiňují.

Ultrazvukové

- Poloha bodů se vypočítává na základě triangulace zdrojů zvukových vln.
- Nevýhodou je odraz vln a tedy zkreslení výsledků.

Inerční

- K zaznamenání pohybu se využívají akcelerometry, které měří zrychlení v měřeném bodě.
- Nevýhodou je degradace přesnosti polohy bodů. Odchylka naměřené polohy bodu od skutečné polohy se s rostoucím časem zvětšuje kvůli nepřesnosti technologie.

Bezmárkrové

- Na tomto principu pracuje zařízení Kinect.
- Toto zařízení snímá prostor a registruje objekty na scéně. Následně z nasnímaných dat generuje hloubkové mapy a detekcí objektů z kamer generuje kostru a pohyb.
- Nevýhodou této metody je nízká přesnost a velké výpočetní nároky.

Hybridní

- Pro zjištění polohy jedince se využívají kombinace výše zmíněných metod. Snahou je eliminovat jejich nevýhody.

2.1.2 Formáty pro uložení dat

Naměřená MOCAP data jsou ukládána v různých formátech. Některé jsou v textové podobě, jiné jsou uložené binárně. Formát ASF/AMC ukládá zvlášť do jednoho souboru popis kostry a do druhého popis pohybu. Podobným formátem je formát BVH, který vše ukládá v jednom souboru. Existuje možnost ukládání dat ve formátu FBX, který do souboru ukládá celou scénu včetně poloh světel případně modelu, který se mapuje na pohyb kostry (Obrázek 2.1). Dále existuj formát C3D, původně vyvinutý pro lékařské potřeby. Soubor obsahuje surová data naměřená systémem, pouze souřadnice markrů umístěných na herci. Známý je i formát TVD, který obsahuje surová data ze zařízení Vicon nebo ze zařízeních jemu podobných.

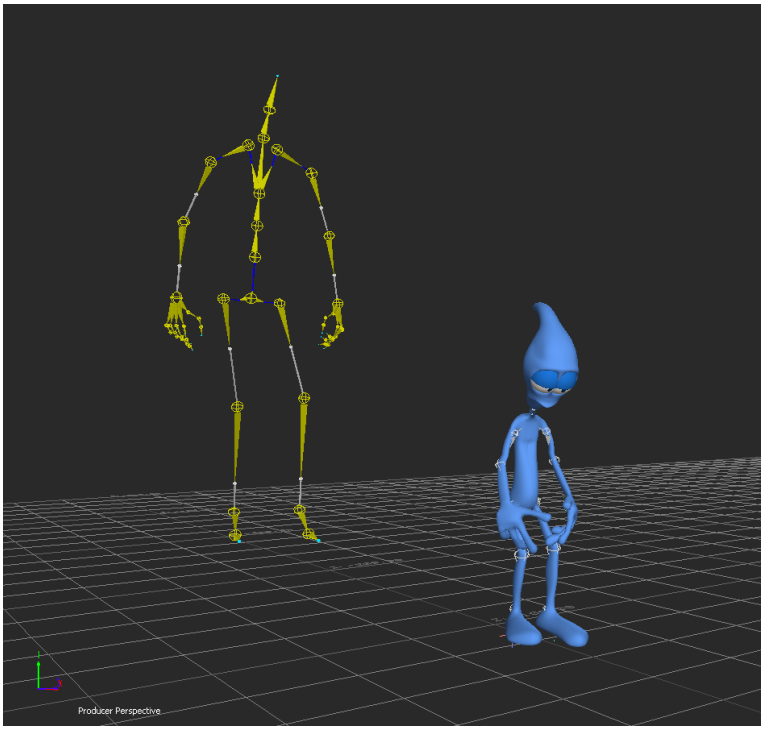
Výhody formátu FBX:

- snadné čtení souboru: Pro formát FBX je veřejně dostupné SKD od Autodesk
- široce používaný formát: Tento formát se velmi často používá ve studiích pracující s MOCAP daty.

Nevýhody formátu FBX:

- podporovoné z velké většiny programy od Autodesk
- pro mé použití je zbytečně složitý a obsahuje nadbytečná data

Vzhledem k výhodám a k tomu, že formát FBX využívá studio, pro které se algoritmus vyvíjí, budu vyvíjet a testovat právě nad daty v tomto formátu.



Obrázek 2.1: Data uložená v FBX

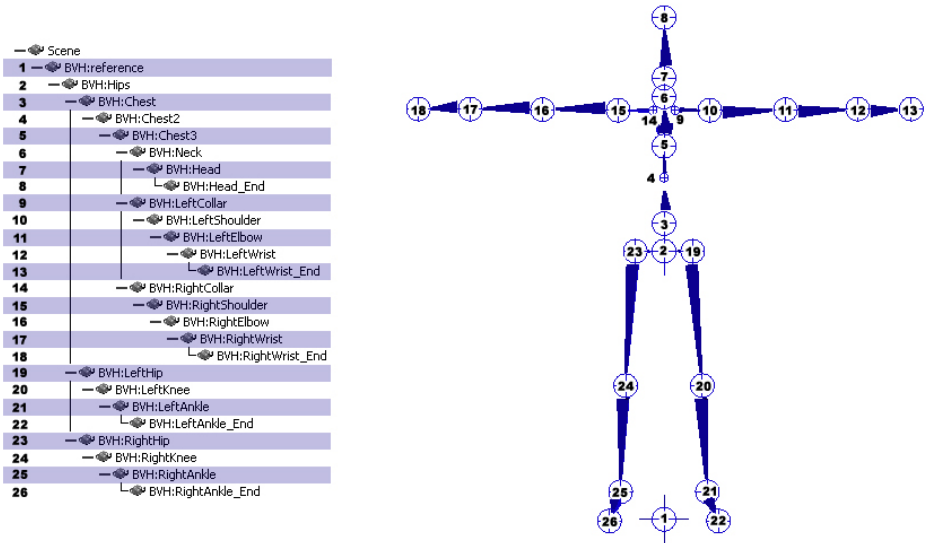
2.1.3 Struktura dat kostry

Kostra je uložena ve stromové hierarchii. Kořenem stromu je standardně místo na páteři v oblasti beder. Potomky kořene jsou pak vyšší bod na páteři a kyčle.

Od kyčlí se větve rozrůstají až k chodidlům. Podle přesnosti měření je často koncovým efektorem pouze palec. Někdy jsou ale zaměřeny i jednotlivé prsty na nohou.

Od páteře vede větev stromu směrem k lopatkám, kde se větví opět třemi potomky. Jedním potomkem je směrem ke krku zakončený efektorem na hlavě. Druzí dva potomci jsou ramena. Ty končí efektory na rukou. Opět zde záleží na přesnosti měření. Některá data končí jedním efektorem, jiná jsou zakončena jednotlivými prsty.

Struktura kostry je nejlépe znázorněna na obrázku 2.2. Jde o strukturu BVH souboru. Jak je ale vidět, uzel `BVH:Hibs` je předkem pro všechny uzly reprezentující klouby kostry.



Obrázek 2.2: Struktura kostry[1]

Kapitola 3

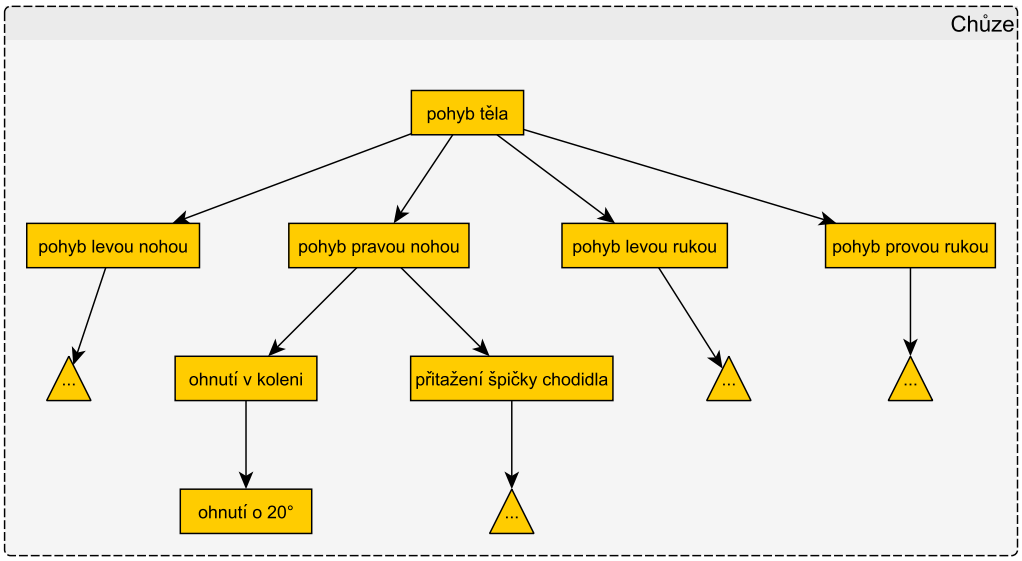
Analýza

3.1 Využití kategorizace pohybu

Ve velkých databázích s naměřenými pohyby je výhodou jednotné umístění. Zároveň se ale v takovém množství dat velmi problematicky vyhledává. Krátký, často jednoslovný popis jako „chůze“ nepopisuje pohyb s dostatečnou přesností, aby se dal mezi dalšími pohyby odlišit. Například chůze muže je jiná než chůze ženy. Chůze s kulháním je také odlišná. Právě k lepšímu odlišení pohybu je zapotřebí automatická kategorizace pohybu.

Pohyb osoby se skládá z menších pohybů. Tyto menší pohyby poté můžeme popisovat v průběhu trvání pohybu a podle toho přesně definovat, o jaký pohyb jde. Obrázek 3.1

S rostoucí hloubkou, jak sestoupíme do diagramu, určíme, jak detailně pohyby rozlišíme mezi sebou. Pokud bychom chtěli rozlišit jdoucí osobu a stojící osobu, která mává, podaří se nám to již na úrovni porovnávání pohybu nohou.



Obrázek 3.1: Diagram pohybu chůze

3.2 Požadavky na kategorizaci pohybu

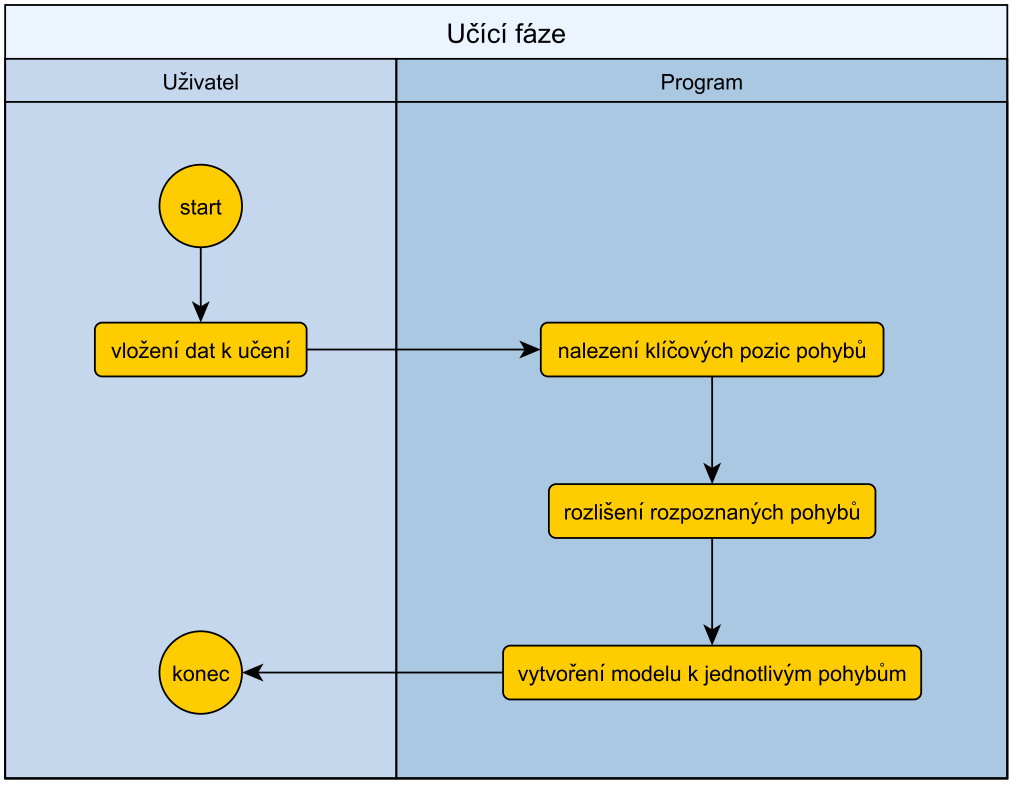
V závislosti na složitosti pohybu je potřeba pohyb stejně podrobně popsat. Takovýto popis musí být správně strukturovaný pro možnost vyhledávání. Důležité je pohyb popsat nejen přesně kvůli hledání detailu, tak i obecně pro zachycení základních znaků. Pro popis pohybu tedy využijeme strukturu ve formátu XML.

XML nám bude sloužit podobně jako popis v ukázkovém diagramu na obrázku 3.1. V nejvyšší úrovni bude definován popis v nejjednodušší formě. Dále s každým sestoupením do podelementu se definovaný pohyb v rodičovském elementu bude zpřesňovat. Tímto způsobem bude možné vyhledávat efektivně jak obecně hledanou chůzi, tak chůzi s máváním dohromady.

3.3 Rozlišení pohybů

Kategorizace pohybů se skládá ze dvou základních částí. První částí je naučit program určité pohyby z předem nadefinovaných a již kategorizovaných vzorků. V této části program musí být schopen načíst data, která po zpracování využije pro rozpoznávání pohybů. Obrázek [3.2]

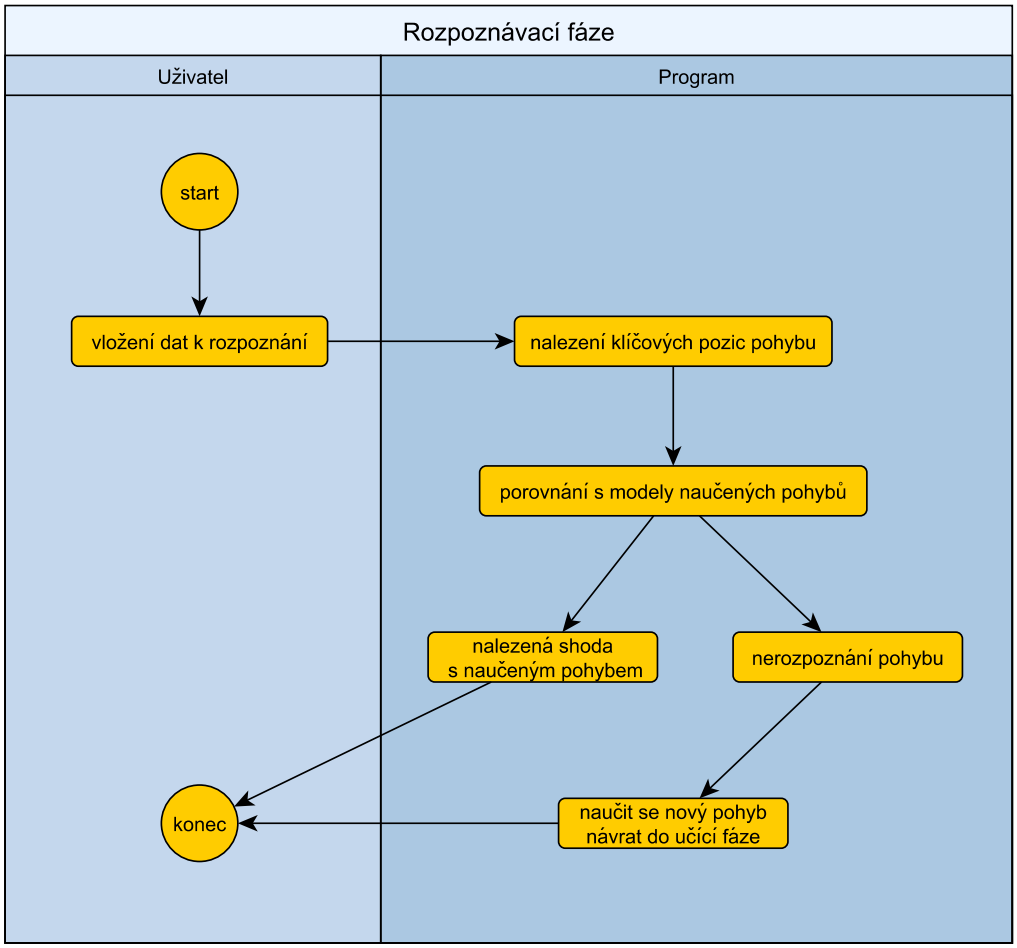
Tím se dostáváme ke druhé části. Ta má za úkol nabídnout uživateli možnost vložit



Obrázek 3.2: Učící fáze programu

soubor s daty o pohybu. Program tato data zpracuje a dále porovná s již naučenými daty. Pokud pohyb není rozpoznaný, vrátí se program se zpracovanými daty zpět do první části

a data zakomponuje jako nový pohyb mezi již naučené. Při dalším běhu programu bude již takový pohyb úspěšně rozpoznán. Obrázek [3.3]



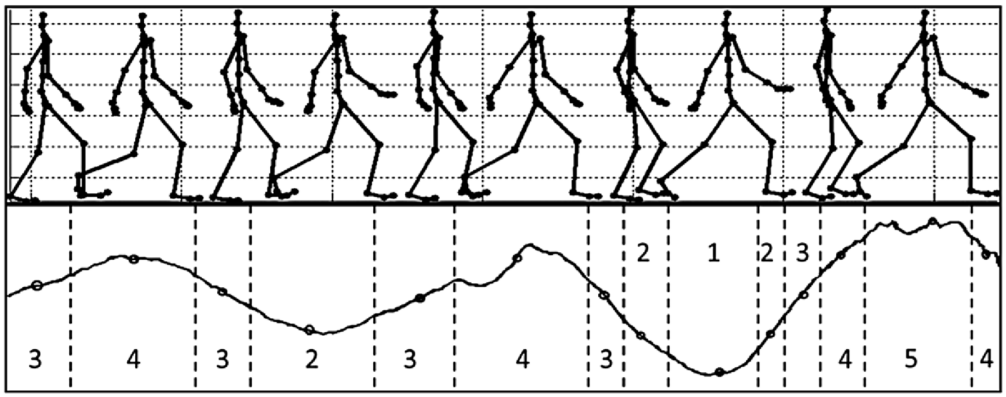
Obrázek 3.3: Rozpoznávací fáze programu

3.3.1 Hledání klíčových poloh

Jednou z možností, jak nalézt klíčové polohy, je použít K-means algoritmus. Inspirací je článek [6]. Tato metoda využívá Euklidovské vzdálenosti mezi jednotlivými snímky. Cílem je minimalizovat vzdálenost mezi jednotlivými snímky a snímkem ideálním pro klíčovou pozici. Předem je nastavena mez δ. Mez je definována jako maximální vzdálenost všech snímku v okolí klíčové polohy. V článku je označováno jako cluster. Pokud vzdálenost mezi klíčovým snímkem je větší než zmíněná δ, cluster se ukončí a vytvoří se nový. Takto nalezené pozice znázorňuje obrázek 3.4.

Druhou možností je nově vymyšlený způsob, jak najít klíčové polohy. Vychází se z principu pohybu. V případě chůze jde o pohyb, kdy se střídají nohy a ruce. Důležité jsou pro nás

tedy pozice, kdy se mění směr pohybu nohou a rukou nebo-li kdy se mění rotace. Pro případ chůze, tedy snímek, kdy je pravá noha vpředu a následně kdy je levá noha v předu. Totéž bude platit i pro ruce. Zde využíváme toho, že data o pohybu nejsou uložena globálně, ale jako velikost rotace a offset. Abychom zamezili detekci vibrací v kostře, musíme si zvolit ještě jednu konstantu τ. Tato konstanta nám bude definovat minimální velikost rotace. Jakoukoliv změnu menší než τ budeme tedy považovat za vibraci.



Obrázek 3.4: Nalezení klíčových poloh pomocí K-means algoritmu

3.3.2 Vytvoření modelu pohybu

Pokud v učící fázi bude pouze jeden zástupce pohybu, tento krok je pro třídu pohybu zbytečný. Jeden zástupce je nevýhodný, protože popisuje pohyb jednoho daného jedince. Chůze jiného člověka může být natolik odlišná, že by program nenašel shodu. Vliv na to má délka kostí a stav svalstva. Abychom pohyb zobecnili, musíme nasnímat více jedinců s daným pohybem a vytvořit model sjednocující pohyb všech.

3.3.3 Zjištění typu pohybu

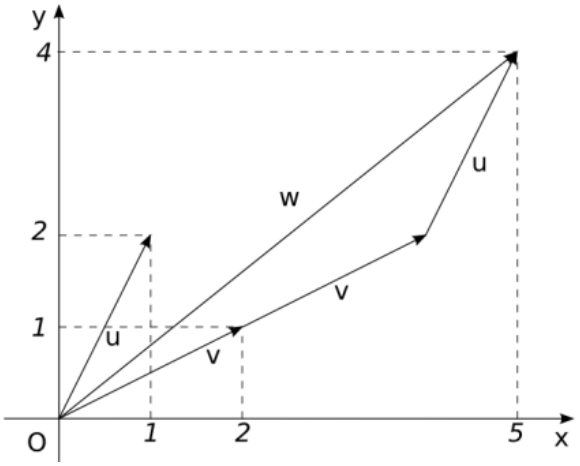
Naměřený pohyb, který budeme chtít kategorizovat, budeme porovnávat s naučenými modely pohybů. Abychom ale měli možnost je porovnat, musíme u neznámého pohybu nalézt klíčové polohy stejně jako v učící fázi. Po nalezení klíčových poloh pohybu dojde k porovnávání.

Každý model má na začátku stejnou pravděpodobnost na výskyt. Postupně, jak se prochází nalezené polohy, některé modely díky vyšší shodě získají vyšší pravděpodobnost. Po projití pohybu bude mít jeden model nejvyšší pravděpodobnost, neboli nejvyšší shodu. Pokud pravděpodobnost bude menší než zvolená mez, kategorizace se označí jako neúspěšná. V takovém případě se pro nový pohyb vytvoří nová třída.

Výsledkem porovnání s modely pohybu získáme úseky s různou kategorizací. Tyto úseky se mohou překrývat. Tato označení jsou následně výstupem, který se uloží do formátu MPEG-7. Při budoucím vyhledávání již nebude nutné pohyb znovu analyzovat, ale bude stačit pouze vyhledávat ve vygenerovaném popisu

3.3.4 Porovnání pohybů

Naučené modely pohybu obsahují vektory posunu a rotací. Abychom mohli porovnat model s novými pohyby, musíme k tomu definovat metriku. Nejlépe potřebujeme ohodnotit novou pózu procentuální shodou s naučeným modelem. To uděláme tak, že srovnáme rotace a posuny příslušných uzlů. Jako metriku tedy zvolíme podobnost vektorů. Obrázek 3.5



Obrázek 3.5: Porovnání vektorů[5]

V programu využívám takovýto vzorec. Výsledkem je hodnota, udávající z kolika procent je vektor podobný naučenému.

$$similarity = 1 - Min\left(\frac{|w - v|}{|w|}, 1\right)$$

3.3.5 Výstup pro formát MPEG-7

Standard MPEG-7 je formát specializovaný pro popis multimediálního obsahu. Výhodou tohoto formátu je možnost definovat si vlastní prvky. Prvky jsou v syntaxi XML.

Po kategorizaci se sekvence uloží do XML struktury. V našem případě se uloží sekvence zvlášť. Pokud bude uživatel hledat složitější pohyby, nejčastěji složené z několika menších, budou se hledat průniky sekvencí, které odpovídají hledanému popisu.

Databáze, které podporují XPath a XQuery tak budou moci v souborech efektivně vyhledávat pohyby, které uživatel popíše. Příklad vytvořené struktury dokumentu na obrázku 3.1.

3.4 Kategorizace pohybu v choreografii

3.4.1 Choreografie

Choreografie se v první řadě zaobírá umístěním tanečníků na scéně. Její snahou je připravit základní plán pohybu a rozmístění. I v největším detailu je pohyb popsán pouze jako přesun

z bodu X do bodu Y pomocí určitého počtu pohybů, např. kroků nebo skoků. Pro naše účely se tedy technika choreografie jako vzor pro popis pohybu nehodí.

3.4.2 Popis standardních tanců

Pomoci nám však může popis standardních tanců. Tanec je díky historii detailně popsán.

- Specifikace kroků a držení těla
- Rozdíl mezi pánskými a dámskými kroky

Waltz

Princip:

- Od valčíku se liší především svým pomalejším tempem a houpavým charakterem pohybu.
- Na konci třetí doby jde tanečník do snížení, z kterého vychází vpřed (dáma vzad) a provádí zdvih, který vrcholí v druhé době.
- Tančí se po obvodu sálu.

Základ:

- Takt: 3/4
- Tempo: 28-32 taktů za minutu
- Postavení: čelem po směru tance
- Držení: uzavřené postavení

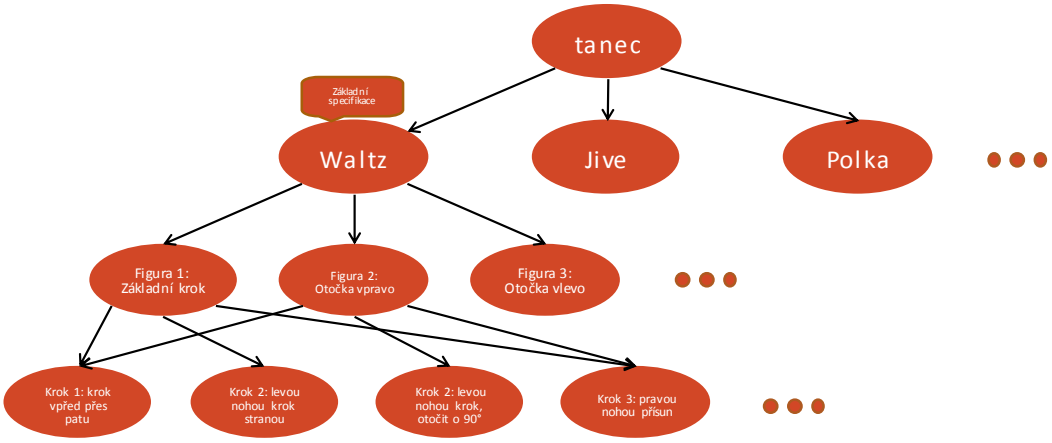
Kroky (pánské):

1. Pravou nohou krok vpřed přes patu (do kroku dívky) se snížením čelem po SMT
2. Levou nohou krok stranou se zdvihem (který začíná již na konci 1. doby) Postavení: čelem po směru tance
3. Pravou nohou přísun k levé noze do vysokého zdvihu (na konci této doby začíná snížení)
4. Druhý takt je totéž na druhou stranu (pán vychází levou nohou vpřed)

3.5 Kategorizace standardních tanců

Standartní tance svým detailním popisem vybízí ke stromovému třídění.

Tanec dělíme podle druhu. U každého druhu tance můžeme definovat základní parametry, tempo, postavení tanečníků atd. Dále v každém tanci můžeme definovat jednotlivé figury. V nejnižší úrovni stromu se nachází jednotlivé kroky. Jak je v diagramu vidět, jednotlivé kroky mohou být stejné pro různé figury. Obrázek 3.6



Obrázek 3.6: Kategorizace standardních tanců

Kód 3.1: Ukázka popisu pohybu. Zdroj [4]

```
<!-- motion description listing -->
<mes:MediaTime>
  <mpeg7:MediaTimePoint>T00:00:00:0F00</mpeg7:MediaTimePoint>
  <mpeg7:MediaDuration>PT1N10F</mpeg7:MediaDuration>
</mes:MediaTime>
<mes:MotionDescriptor>
  <mes:Action>Walking knight</mes:Action>
  <mes:MotionClassDescriptor>
    <mes:Anotation>Walking biped</mes:Anotation>
      <mes:ClassName>WALK</mes:ClassName>
      <mes:id>W001</mes:id>
      <mes:MotionDetails>
        <mes:item><mes:name>steps</mes:name>
                  <mes:value>6</mes:value>
        </mes:item>
        <mes:item><mes:name>WalkingGateStatus</mes:name>
                  <mes:value>right leg defect</mes:value>
        </mes:item>
        <mes:item><mes:name>FigureStatus</mes:name>
                  <mes:value>bended forward</mes:value>
        </mes:item>
      </mes:MotionDetails>
      <mes:PartDecomposition>
        <mes:item><mes:name>hands</mes:name>
                  <mes:value>bended in elbow</mes:value>
        </mes:item>
        <mes:item><mes:name>hands</mes:name>
                  <mes:value>sword in right hand</mes:value>
        </mes:item>
    </mes:PartDecomposition>
    <mes:MotionSpeed>1.0</mes:MotionSpeed>
  </mes:MotionClassDescriptor>
</mes:MotionDescriptor>
```

Kapitola 4

Návrh řešení

4.1 Analýza úzkých míst návrhu

4.1.1 Problematika rozpoznání pohybu

Rozpoznání pohybu provází několik úskalí

- Velké množství dat
- Timewarp
- Kmitání

Množství dat
Naměřená data obsahují informace o polohách desítky bodů. Polohy jsou ve velké přesnosti. Aby se při záznamu zamezilo aliasingu, je navíc nutné zaznamenat běžný pohyb velice často 120 až 160 snímků za sekundu (fps). Pokud by se jednalo o rychlé pohyby, snímkování se může zvýšit až na cca. 1000fps.

Timewarp
Tento jev vzniká při záznamu stejného pohybu jinou snímkovací rychlostí. Výsledkem je, že stejný pohyb máme nasnímaný různým počtem snímků. Pokud bychom porovnávali snímky první s první, druhý s druhým atd., ve výsledku bychom porovnali celý pohyb jednoho záznamu s částí pohybu druhého záznamu.

Kmitání
Velká přesnost záznamu je důležitá při snímání detailních pohybů. S velkou přesností ale zároveň vzniká efekt kmitání. Pozice bodu, který by měl mít stále stejnou pozici, tak ve výsledku nemá stejnou hodnotu, ale mění se v blízkém okolí. Toto kmitání je zaznamenáno jako pohyb, kterého se snažíme zbavit.

4.1.2 Rozdílnost koster

Rozdílná anatomie člověka

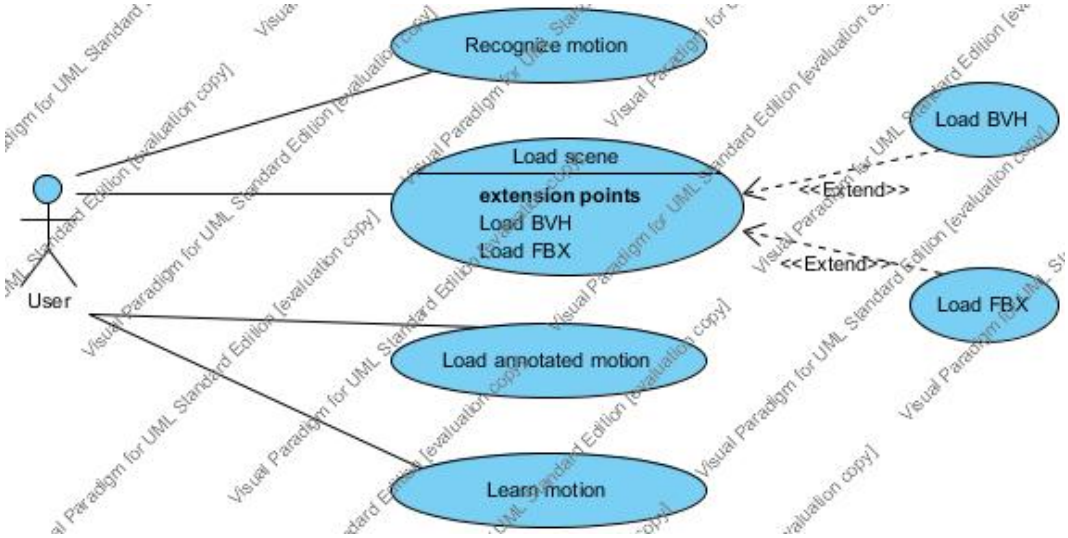
Dalším úskalím jsou různé druhy koster. Při získávání testovacích dat jsem narazil na rozdílné způsoby uložení pohybových dat. Tato data se vztahují k jednotlivým kloubům. Tím, jak se od sebe kostry liší, jsou tím ovlivněna i data. Snímané postavy nejsou stejné, a tak se délka jednotlivých kostí liší.

Postproccessing nasnímaných dat

Další rozdíl je při samotném zpracování dat. Pouhá snímání anatomie obsahují pouze ty nejnutnější klouby. V podstatě pouze zjednodušenné klouby, které obsahuje skutečná kostra člověka. Pokud se ale pohyb používá pro animaci, jsou do kostry přidávány další klouby a kosti. Tyto klouby obsahují také animaci, ale ta je využíváná hlavně k animaci mapovaného modelu.

4.2 Use cases

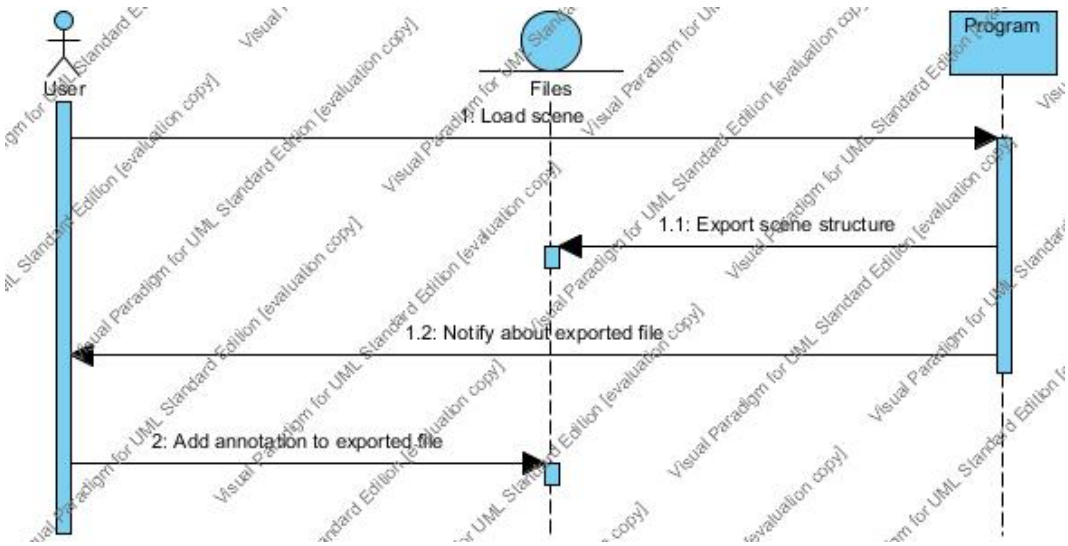
Program nabízí několik funkcí, které uživatel používá k učení a rozpoznávání pohybů. (obrázek 4.1)



Obrázek 4.1: Use case podporované programem

4.2.1 Načtení souboru scény a export struktury

První, co uživatel musí udělat, je vyexportovat strukturu souboru obsahující pohyb. Tento úkol je důležitý hlavně pro soubor FBX, který obsahuje celou scénu uloženou ve stromové struktuře. Program tedy vyexportuje tuto stromovou strukturu. (obrázek 4.2)



Obrázek 4.2: Načtení scény

4.2.2 Výběr důležitých částí pohybu

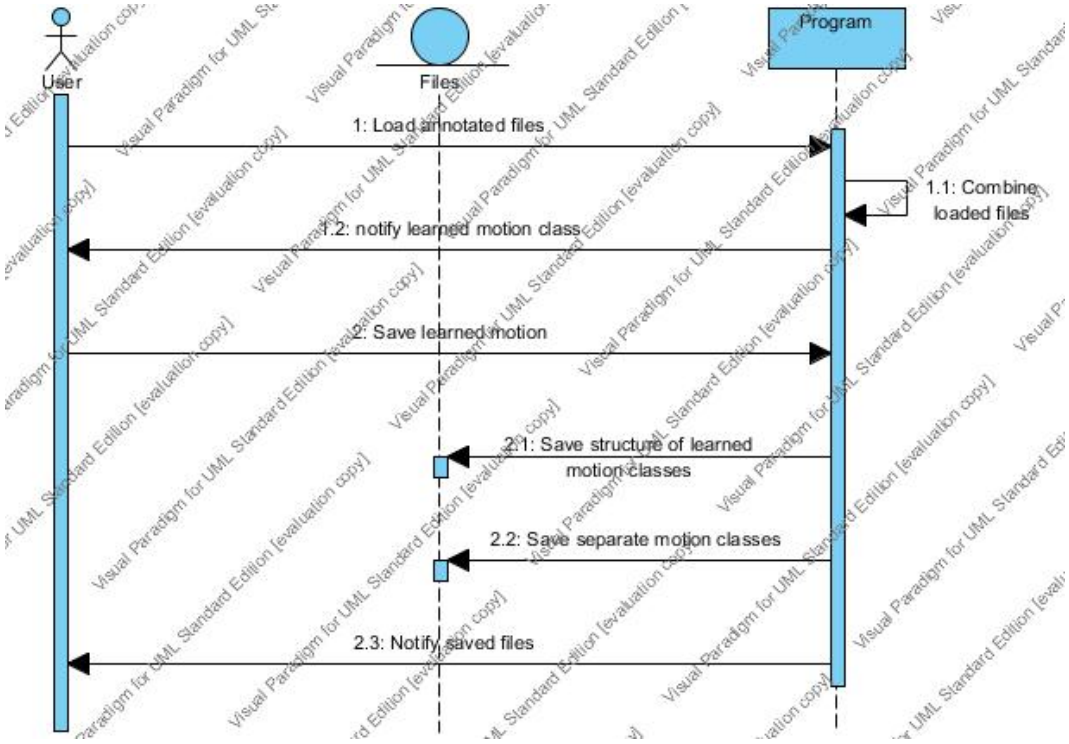
Než uživatel může spustit fázi učení pohybů, musí označit uzly scény, které obsahují pro nás zajímavé animační křivky. Udělá to tak, že zkopírovat uzly xml do tagu `annotation`. Důležité je ještě označit správně názvy uzlů v atributu `annotation` v jednotlivých elementech `node`. Tyto názvy jsou důležité pro sjednocení označení animačních křivek. Názvy uzlů nejsou mezi formáty ustálené a tak je každý popisuje jinak.

4.2.3 Načtení označených částí scény a naučení pohybu

K naučení pohybu stačí vložit upravený soubor a zadat název pohybu. Soubor s označenými uzly obsahuje i cestu k původnímu souboru scény. Označené uzlů se vyhledají ve scéně a načtou se příslušné animační křivky. Do konzole program v průběhu výpočtu vypíše informace získaná učením. (obrázek 4.3)

4.2.4 Rozpoznání pohybu

Jakmile jsou naučeny pohyby, může uživatel spustit rozpoznávání pohybů. Pro rozpoznání je nutné opět v souboru označit uzly obsahující pohyb, viz. sekce 4.2.2. Uživatel je v průběhu porovnávání informován výsledcích porovnání jednotlivých tříd. Nakonec program do konzole vypíše výsledek rozpoznávání. (obrázek 4.4)



Obrázek 4.3: Naučení pohybu

4.3 Model tříd

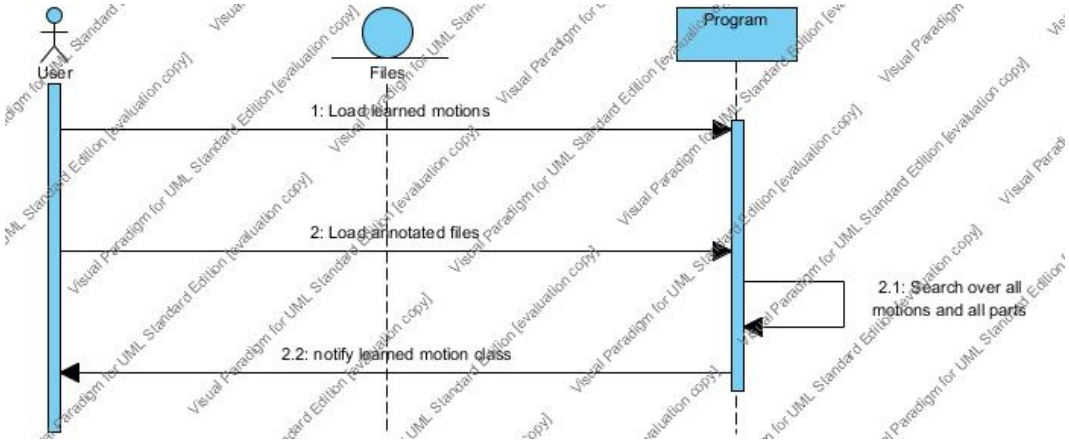
4.3.1 Třídy dat scén

Jelikož jsou pohybová data uložená v různých formátech, bylo nutné vytvořit si vlastní datové struktury. Do těchto struktur se exportují data kostry a pohybu. Práce s daty je poté vždy stejné a není nutné duplikovat kód. V případě že by program byl nucen načítat pohyb z dalších formátů, jediné co bude potřeba doimplementovat, bude parser na tato data.

Při načtení pohybu se vytvoří objekt scény obsahující kostru postavy a odděleně od níc objekt pohybu s animačními křivkami. Tato scéna poté umožňuje vyexportovat důležité snímky. Dále umožňuje uložit data do souboru v čitelné formě. Tuto funkci jsem využíval pro studium dat uložených v původním souboru pohybu. (Obrázek 4.5)

4.3.2 Třídy pro učení

Během učení se vytváří kontejnej, obsahující jednotlivé třídy pohybů. Každá z tříd v sobě ukládá pohybová data, informace o velikosti kostí a důležitost jednotlivých animačních křivek.

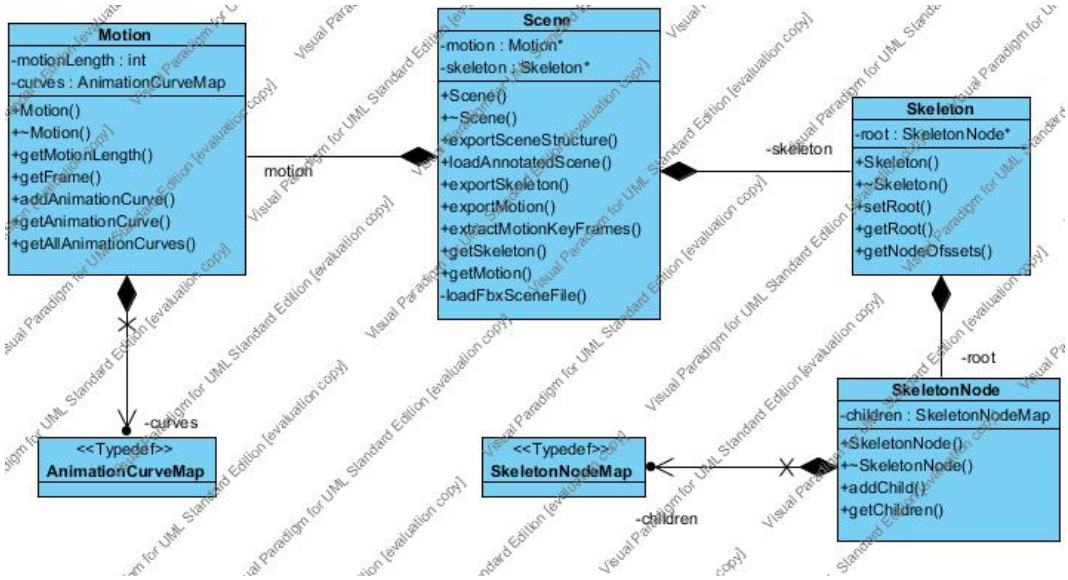


Obrázek 4.4: Rozpoznání pohybu

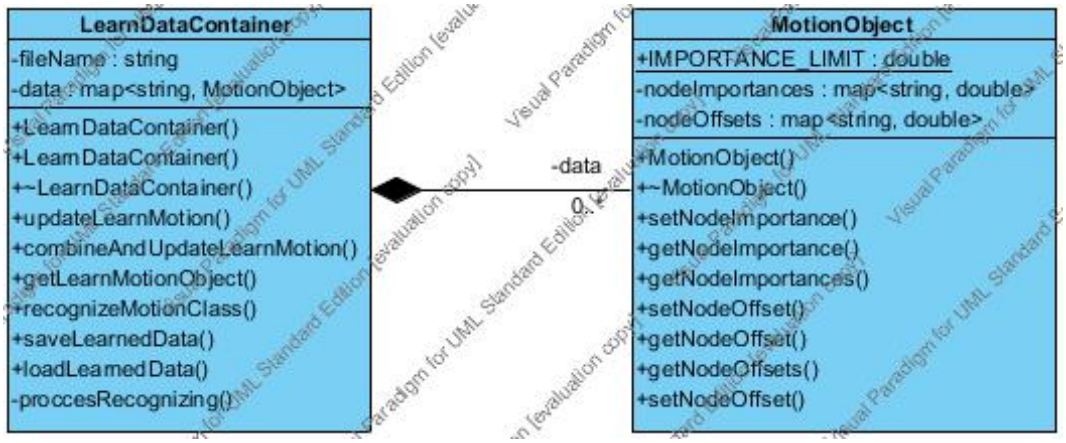
Tento kontejner zaštiťuje veškerou další práci s naučenými daty a funkce pro rozpoznání pohybu. Případě učení dokáže skombinovat vložené pohyby jedné třídy pohybu. V těchto pohybech najde nejdůležitější animační křivky a ze všech dat vytvoří jeden vzorek rerezentující danou třídu. (Obrázek 4.6)

4.3.3 Třídy pro rozpoznávání

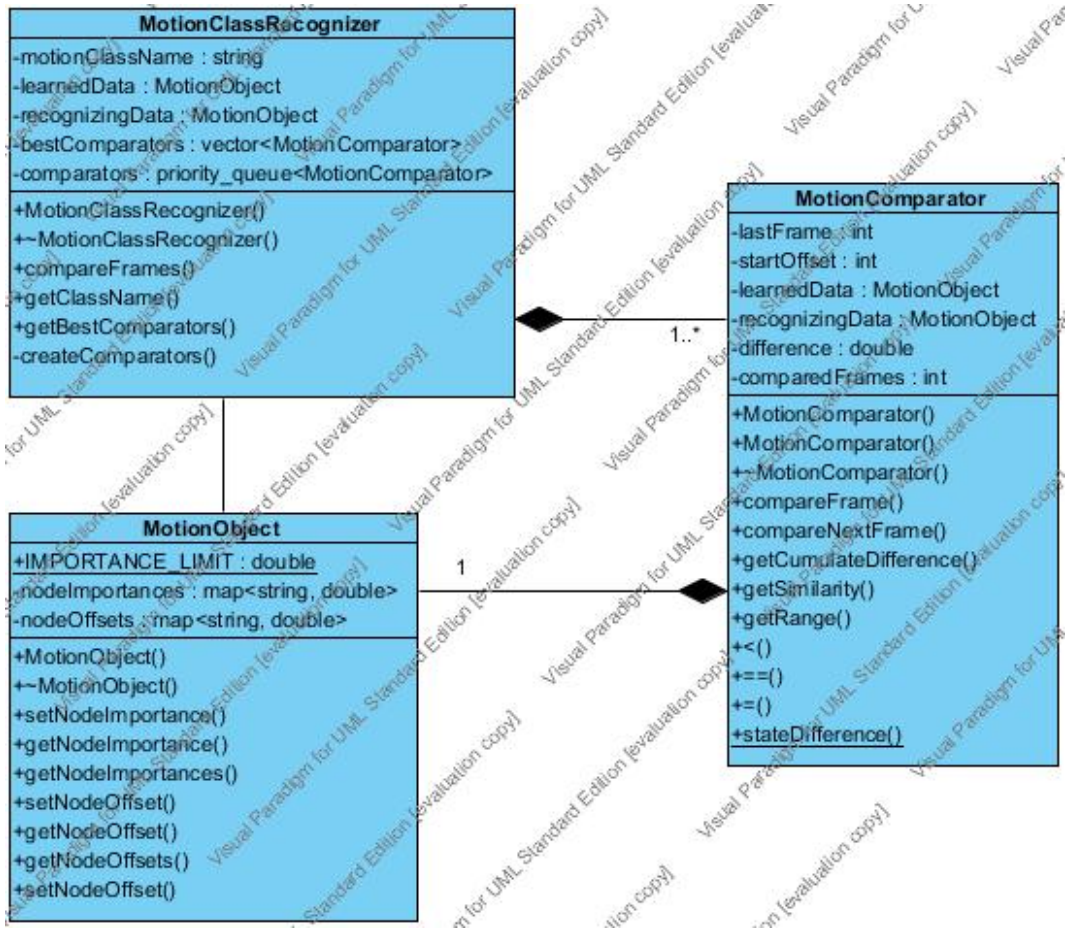
Pro rozpoznání pohybu se též vytvářejí struktury pro porovnávání. Kontejner obsahující naučená data vytváří třídy `MotionClassRecognizer` zorpovědné za porovnání třídy pohybu. Každá z nich si poté vytváří jednotlivá ok `MotionComparator`, které teprve porovnávají jednotlivé klíčové pozice a vektory definující pohybu.(Obrázek 4.7)



Obrázek 4.5: Model uložení scény



Obrázek 4.6: Model uložení scény



Obrázek 4.7: Model pro rozpoznávání

Kapitola 5

Prototyp

Tento program je pouze prototyp. Využije se pro následné větší testování a rozvíjení funkcí.

Schopností, ale zároveň i nevýhodou je váhování animačních křivek. Je tím velice ovlivněný výsledek rozpoznávání. Snižuje se schopnost rozpoznání detailů. Pohyby malých kostí se téměř neregistrují.

5.1 Spuštění programu

Program se spouští se několika parametry.

MotionClassifiction.exe **L S F []**

parametr L	udává, jestli se má načíst naučená data [0, 1]
parametr S	udává, jestli se má uložit naučená data [0, 1]
parametr F	udává, která funkce se má použít
parametr []	na základě vybrané funkce se mění parametry, které se mají zadat

Export struktury scény F=0 Následuje jediný parametr s cestou k souboru, ze kterého se má vyexportovat struktura.

Naučení třídy pohybu F=1 Následuje parametr s názvem třídy, kterou se bude program učit. Poté je vstupem neomezený počet parametrů, kterými jsou cesty k souborům s anotací struktury.

Rozpoznání pohybu F=2 Následuje jediný parametr s cestou k souboru, který se má rozpoznat

5.2 Funkce

5.2.1 Export struktury scény

MotionClassifiction.exe **0 0 0 "./walk.bvh"**

Jelikož soubory s pohyby obsahují více dat než jen data animačních křivek, které nás zajímají, je nutné programu označit ta správná data. K tomu program vyexportuje strukturu scény.

Do stejné složky, ve které je uložený soubor s daty o pohybu, uloží XML a DTD soubor. XML má uloženou strukturu v tagu `<structure>`. Uživatel musí vybrat uzly, které ho zajímají, a zkopírovat je do tagu `<annotation>`. Tím je uživatel označí pro další funkce programu.

5.2.2 Naučení třídy pohybu

MotionClassifiction.exe **1 1 1 "./walk.xml"**

Program načte již naučené pohyby. Následně zkombinuje jednotlivé soubory s pohyby do jedné třídy a uloží je. Pokud již existuje stejná třída, bude přepsána nově naučenou verzí. Celková struktura se uloží do `./learned/learned_data.xml` a data jednotlivých tříd do oddělených souborů.

5.2.3 Rozpoznání pohybu

MotionClassifiction.exe **1 0 2 "./walk2.xml"**

Program načte již naučené pohyby. Následně načte soubor s anotací. Pokud je vložen původní soubor pohybu, program automaticky vyexportuje strukturu (funkce 1, kapitola 5.2.1), počká, až uživatel vytvoří v souboru anotaci, a tento soubor načte. Pohyb porovná s naučenými třídami a do konzole vypíše třídy, které v pohybu našel.

Kapitola 6

Implementace

Program je napsaný v jazyku C++. Pro implementaci jsem využíval školní verzi vývojového prostředí Visual Studio 2013. V návrhu implementace mi pomohl program Visual Paradigm for UML 11.0, ve kterém jsme navrhl datové struktury.

6.1 Stavební prvky řešení

Program využívá 2 knihovny třetích stran.

6.1.1 FBX SDK [2]

Licence Creative commons public license

Knihovna vyvinuta firmou Autodesk. Tato firma si pro svá prostředí vytvořila vlastní formát FBX. Podporuje i načítání souborů ve formátu BVH. Knihovna je volně stažitelná pro rychlejší rozšíření podpory jejich formátu.

Využití Načítání souborů FBX a BVH.

6.1.2 PugiXML [3]

Licence MIT license

Knihovna složí ke čtení a zapisování dat do soubooru XML. Nepodporuje prási s DTD soubory.

Využití Export naučených pohybů a struktury scén obsahující pohyb.

Při implementaci jsem vycházel z článku zadaného vedoucím práce. Článek popisuje základní algoritmus pro učení programu, algoritmu rozpoznávání se nevěnuje jen z části. Proto jsem rozpoznávání implementoval svépomocí a ze znalostí získaných během studia.

6.2 Algoritmy pro učení

Při implementaci učení jsem vycházel z interpolace vektorů, jelikož se při učení využívá více vzorků pohybu. Bylo nutné vzorky průměrovat do jednoho.

Průměr rotace Průměr rotace jsem spočítal jako:

$$r = \frac{r_1 + r_2}{2}$$

Výsledná rotace je pouze lineární interpolace hodnot v každé z os.

Průměr translace Průměr translace jsem spočítal jako:

$$t_{tmp} = \frac{t_1 + t_2}{2}$$

$$t = \frac{t_{tmp}}{|t_{tmp}|} * (\frac{|t_1| + |t_2|}{2})$$

Výsledná translace má lineárně interpolované hodnoty v každé z ose a výsledný vektor je normalirovaný a vynásobený průměrnou velikostí původních vektorů.

Následně se přes všechny vzorky průměrují velikosti kostí.

Během kombinování snímků ze vzorků se jednotlivé rotace a translace porovnávají. Na zádladě podobnosti se vypočítává důležitost animačních křivek. Čím více jsou tranformace ve vzorcích podobnější, tím je důležitost vyšší. Následně se do hodnoty ještě započítá velikost kosti. Bez této důležitosti klesá přesnost rozpoznávání pohybu. Rotace malých kostí jsou často velké. Pronásobením s vypočítanou hodnotou jejich vliv na výsledek klesá.

6.3 Uložení dat

Naučená data se při ukončení programu ukládají pro možné příští načtení. Struktura je definována pomocí DTD. Jednotlivé třídy pohybu jsou stromově strukturované. Tím je možné definovat složitější typy pohybu. Každá třída pohybu se skládá z dalších tříd nebo objektů. Každý objekt poté v externím souboru obsahuje naučená data.

Objekty pohybu obsahují 3 druhy informací. Nejdříve je uložena důležitost animačních křivek a velikost kostí. Obě informace se užívají pro váhování transformací, a tedy z kolika procent se má transformace počítat do daného pohybu. Poté následují samotné snímky pohybu.

Důležitost animační křivky se vypočítává na základě shody trasformací mezi více vzorky pohybů. V případě více vzorků pohybu se porovnávají extrahované snímky. Jednotlivé snímky se porovnávají a křivky, které se ve výsledku nejvíce shodují, mají i nejvyšší procento důležitosti.

Velikost kostí poté ovlivňují vypočítanou důležitost. Koncové uzly jsou tímto způsobem utlumeny, aby při porovnávání pohybů tolik neovlivňovaly výsledek.

Snímky jsou indexovány od nuly a ke každému uzlu je zapsán posun rotace a zvětšení, vše ve 3D.

Kód 6.1: Struktura dat naučených pohybů

```
<?xml version="1.0"?>
<!DOCTYPE motion_class SYSTEM "learned_data.dtd">
<motion_class>
  <motion_object name="jump" length="18" path="jump" />
  <motion_object name="run" length="12" path="run" />
  <motion_class name="walk" cyclic="true">
    <motion_object name="step_left" length="37"
        path="step_left" />
    <motion_object name="step_right" length="37"
        path="step_right" />
  </motion_class>
</motion_class>
```

Kód 6.2: Soubor naučeného pohybu

```
importances 38
BVH:Head : 0.123844
BVH:Head_End : 0
BVH:Hips : 0
BVH:LHipJoint : 0
BVH:LThumb : 0
BVH:LThumb_End : 0
...
offsets 38
BVH:Head : 1.67929
BVH:Head_End : 1.68654
BVH:Hips : 0
BVH:LHipJoint : 0
BVH:LThumb : 0
BVH:LThumb_End : 0.612567
...
frame 0 38
BVH:Head : 5.66202 -0.173462 4.07891 -1 -1 -1 0 0 0
BVH:Head_End : 0 0 0 -1 -1 -1 0 0 0
BVH:Hips : 6.63723 -10.269 4.22187 3.56872 16.5497 -30.3696 0 0 0
BVH:LHipJoint : 0 0 0 -1 -1 -1 0 0 0
BVH:LThumb : 0.162038 44.7626 -12.6741 -1 -1 -1 0 0 0
BVH:LThumb_End : 0 0 0 -1 -1 -1 0 0 0
...
frame 1 38
BVH:Head : 5.63861 0.224069 4.10304 -1 -1 -1 0 0 0
BVH:Head_End : 0 0 0 -1 -1 -1 0 0 0
BVH:Hips : 6.53017 -9.66767 3.54771 3.39354 16.623 -29.0361 0 0 0
BVH:LHipJoint : 0 0 0 -1 -1 -1 0 0 0
BVH:LThumb : 0.0259062 44.6213 -12.3835 -1 -1 -1 0 0 0
BVH:LThumb_End : 0 0 0 -1 -1 -1 0 0 0
...
frame 2 38
BVH:Head : 5.65009 -0.6867 3.97656 -1 -1 -1 0 0 0
BVH:Head_End : 0 0 0 -1 -1 -1 0 0 0
BVH:Hips : 5.23627 -10.9539 4.08236 3.15264 16.6251 -27.5474 0 0 0
BVH:LHipJoint : 0 0 0 -1 -1 -1 0 0 0
BVH:LThumb : 0.543156 43.8136 -13.3209 -1 -1 -1 0 0 0
BVH:LThumb_End : 0 0 0 -1 -1 -1 0 0 0
...
```

Kapitola 7

Testování

7.1 Hardware

Celý test byl prováděn na notebooku s těmito parametry:

Tabulka 7.1: Parametry hardwaru

CPU	Intel Core i7-3517U
GPU	Intel HD Graphics 4000
RAM	4GB

7.2 Testovací data

Testovací sadu základních pohybů (běh, skok a chůzi) jsem obohatil ještě o tanec (salsa), kopnutí, zvednutí a položení míče a stání.

Data jsem volil tak, aby bylo možné alespoň z části porovnat výsledky se zadaným článkem.

7.3 Učení pohybů

Pro každou třídu pohybu bylo potřeba získat několik vzorků pohybu, aby bylo možné zobecnit pohyb. V takovém množství byl problém získat data ve formátu FBX. Abych mohl tedy testovat tento projekt, získal jsem data alespoň ve formátu BVH. Naštěstí je možné tento formát načítat stejnou knihovnou jako FBX.

Na následujících dat je zřejmé, že se podařilo snížit důležitost menších kostí. Ve výpisech důležitostí jsou jen nejdůležitější kosti. Tedy s hodnoutou důležitosti větší než 0,2.

7.3.1 Skok

Doba učení: 1.771s

Počet scén: 10

Průměrná délka pohybu: 581

Herec skočí dopředu oběma nohama zároň.

Tabulka 7.2: Důležitost kloubů - skok

název kloubu	důležitost [%]
LeftArm	0.383928
LeftFoot	0.650265
LeftForeArm	0.493968
LeftLeg	0.496816
RightArm	0.382567
RightFoot	0.639087
RightForeArm	0.566833
RightLeg	0.556138

7.3.2 Kopnutí

Doba učení: 0.546s

Počet scén: 4

Průměrná délka pohybu: 340

Herec se rozejde a kopne pravou nohou.

Tabulka 7.3: Důležitost kloubů - kopnutí

název kloubu	důležitost [%]
LeftArm	0.39299
LeftFoot	0.483246
LeftForeArm	0.449197
LeftLeg	0.508934
LeftToeBase	0.224793
LeftUpLeg	0.230916
RightArm	0.458164
RightFoot	0.493561
RightForeArm	0.44083
RightLeg	0.545808
RightUpLeg	0.215324

7.3.3 Zvednutí míče

Doba učení: 0.779s

Počet scén: 4

Průměrná délka pohybu: 557

Herec se předehne a pravou rukou zvedne míč. Zároveň s tím zvedá levou nohu kvůli rovnováze.

Tabulka 7.4: Důležitost kloubů - zvednutí míče

název kloubu	důležitost [%]
Head	0.208848
LeftArm	0.397955
LeftFoot	0.745236
LeftForeArm	0.509095
LeftLeg	0.830713
LeftUpLeg	0.218202
RightArm	0.379825
RightFoot	0.685897
RightForeArm	0.588145
RightLeg	0.792682
RightUpLeg	0.221847

7.3.4 Položení míče

Doba učení: 0.738s

Počet scén: 4

Průměrná délka pohybu: 516

Herec popojde a položí míč na zem.

7.3.5 Běh

Doba učení: 0.811s

Počet scén: 11

Průměrná délka pohybu: 140

Herec běží rovně. Vybíhá pravou nohou.

Tabulka 7.5: Důležitost kloubů - položení míče

název kloubu	důležitost [%]
LeftArm	0.320493
LeftFoot	0.6216
LeftForeArm	0.290315
LeftLeg	0.721155
LeftUpLeg	0.238443
RightArm	0.28839
RightFoot	0.493877
RightForeArm	0.309488
RightLeg	0.749945
RightUpLeg	0.207701

Tabulka 7.6: Důležitost kloubů - běh

název kloubu	důležitost [%]
LeftArm	0.349493
LeftFoot	0.214044
LeftForeArm	0.26805
LeftLeg	0.417286
RightArm	0.347124
RightForeArm	0.405209
RightLeg	0.450183

7.3.6 Salsa

Doba učení: 5.71s

Počet scén: 10

Průměrná délka pohybu: 2076

Herec tančí vždy stejnou sestavu několika figur složených z otoček, výměn a základních kroků.

Tabulka 7.7: Důležitost kloubů - salsa

název kloubu	důležitost [%]
LeftArm	0.251722
LeftFoot	0.258353
LeftForeArm	0.219474
LeftLeg	0.508903
RightArm	0.271263
RightFoot	0.385621
RightForeArm	0.248263
RightLeg	0.468359

7.3.7 Stání

Doba učení: 1.291s

Počet scén: 3

Průměrná délka pohybu: 1362

Herec stojí a lehce přešlapuje. V jednom ze vzorků pohybuje rukama.

Tabulka 7.8: Důležitost kloubů - stání

název kloubu	důležitost [%]
LeftArm	0.411165
LeftForeArm	0.5579
LeftLeg	0.527922
RightArm	0.379787
RightForeArm	0.668144
RightLeg	0.431076

7.3.8 Chůze

Doba učení: 1.25s

Počet scén: 9

Průměrná délka pohybu: 349

Herec jde několik kroků rovně.

Tabulka 7.9: Důležitost kloubů - chůze

název kloubu	důležitost [%]
LeftArm	0.321022
LeftFoot	0.278171
LeftForeArm	0.382838
LeftLeg	0.436991
RightArm	0.35627
RightFoot	0.23979
RightForeArm	0.44192
RightLeg	0.446264

7.4 Rozpoznávání pohybů

Pro ukázku rozpoznávání jsem se vybral vzorek chůze.

5 6frame

7.4.1 Porovnání - skok

Velikost třídy: 22

Počet vytvořeních porovnávacích oken: 66

Čas porovnání: 6.959s

```
best comparators:
the others:
    similarity = 0.379569
    similarity = 0.379472
    similarity = 0.378604
    similarity = 0.378082
    similarity = 0.347146
    similarity = 0.238197
    similarity = 0.3141
    similarity = 0.376425
    similarity = 0.375532
    similarity = 0.345696
    ...
```

7.4.2 Porovnání - kopnutí

Velikost třídy: 24

Počet vytvořeních porovnávacích oken: 68

Čas porovnání: 7.522s

```
best comparators:
the others:
    similarity = 0.379123
    similarity = 0.379109
    similarity = 0.378497
    similarity = 0.289545
    similarity = 0.378144
    similarity = 0.378016
    similarity = 0.377934
    similarity = 0.350699
    similarity = 0.377633
    similarity = 0.377143
    ...
```

7.4.3 Porovnání - zvednutí míče

Velikost třídy: 14

Počet vytvořeních porovnávacích oken: 50

Čas porovnání: 3.302s

```
best comparators:
the others:
    similarity = 0.412641
    similarity = 0.412641
    similarity = 0.367433
    similarity = 0.252011
    similarity = 0.366819
    similarity = 0.366705
    similarity = 0.366619
    similarity = 0.365986
    similarity = 0.365682
    similarity = 0.312528
    ...
```

7.4.4 Porovnání - položení míče

Velikost třídy: 20

Počet vytvořeních porovnávacích oken: 64

Čas porovnání: 6.145s

```
best comparators:
the others:
    similarity = 0.364803
    similarity = 0.362364
    similarity = 0.290651
    similarity = 0.361458
    similarity = 0.361453
    similarity = 0.326813
    similarity = 0.360216
    similarity = 0.359951
    similarity = 0.359171
    similarity = 0.358941
    ...
```

7.4.5 Porovnání - běh

Velikost třídy: 13

Počet vytvořeních porovnávacích oken: 57

Čas porovnání: 3.077s

```
best comparators:
the others:
    similarity = 0.257862
    similarity = 0.256453
    similarity = 0.256371
    similarity = 0.256054
    similarity = 0.254481
    similarity = 0.254277
    similarity = 0.254147
    similarity = 0.25345
    similarity = 0.253059
    similarity = 0.25188
    ...
```

7.4.6 Porovnání - salsa

Velikost třídy: 21

Počet vytvořeních porovnávacích oken: 65

Čas porovnání: 6.423s

```
best comparators:
the others:
    similarity = 0.254368
    similarity = 0.129901
    similarity = 0.253684
    similarity = 0.215732
    similarity = 0.253077
    similarity = 0.252524
    similarity = 0.252381
    similarity = 0.252189
    similarity = 0.0748605
    similarity = 0.250923
    ...
```

7.4.7 Porovnání - stání

Velikost třídy: 4

Počet vytvořeních porovnávacích oken: 52

Čas porovnání: 0.75s

```
best comparators:
the others:
    similarity = 0.355787
    similarity = 0.353343
    similarity = 0.35278
    similarity = 0.351968
    similarity = 0.35139
    similarity = 0.349475
    similarity = 0.346229
    similarity = 0.345123
    similarity = 0.34441
    similarity = 0.343332
    ...
```

7.4.8 Porovnání - chůze

Velikost třídy: 41

Počet vytvořeních porovnávacích oken: 85

Čas porovnání: 17.265s

```
best comparators:
    similarity = 0.524246
    similarity = 0.521383
    similarity = 0.519321
    similarity = 0.515647
    similarity = 0.514035
    similarity = 0.509873
    similarity = 0.508433
    similarity = 0.501214
    similarity = 0.500073
the others:
    similarity = 0.485686
    similarity = 0.413927
    similarity = 0.0403586
    similarity = 0.340177
    similarity = 0.318794
    similarity = 0.217721
```

```
similarity = 0.295774
similarity = 0.41281
similarity = 0.244851
similarity = 0.317822
...
```

7.5 Testy

Jelikož aplikace obsahuje více funkcionalit, provedl jsem několik testů, které mají zjistit smysluplnost výpočtů.

1. Velikost naučených pohybů
2. Rozpoznání částí těla důležitých pro daný pohyb
3. Rozpoznání pohybu

7.6 Výsledky testování

7.6.1 Test 1

Z dat v tabulce 7.10 lze vyčíst, že pohyby, které jsou nejvíce statické, jsou nejmenší. Zvláštností je chůze. Výsledek je takto rozdílný jen kvůli tomu, že vzorky pro učení chůze byly 3krát delší než vzorky běhu.

Tabulka 7.10: Velikosti tří pohybu

Salsa	21
Skok	22
Běh	13
Chůze	41
Kopnutí	24
Zvednutí míče	14
Položení míče	20
Stání	4

7.6.2 Test 2

Rozpoznávání částí těla důležitých pro pohyb byl velice úspěšný. Algoritmus úspěšně odfiltroval animační křivky kloubů, které nejsou v pohybech důležité.

V případě chůze a dalších pohybů správně vybral nejdůležitější klouby a těm přiřadil nejvyšší hodnotu. Jedinou nevýhodou je, že při chůzi přiřadil vyšší prioritu i rukám. To je ale způsobené střídáním rukou. Algoritmus tedy pracoval dobře. Snížení priority rukou je za úkol uživatele. Ten má rozhodnout, že i když se ruce také shodují při chůzi, jejich důležitost je minimální.

7.6.3 Test 3

Výstupem jsou jednotlivá porovnávací okna, která mají úspěšnost vyšší než definovaný limit. Stanovil limit na hodnotu `0.5`. Po průniku oken vyšlo, že program rozpoznal chůzi ve snímcích 8 až 56. Všechny ostatní třídy, se kterými se pohyb porovnával, měly úspěšnost porovnání nižší než limit.

```
walk from: 8 to: 56
walk from: 9 to: 56
walk from: 10 to: 56
walk from: 11 to: 56
walk from: 12 to: 56
walk from: 13 to: 56
walk from: 14 to: 56
walk from: 15 to: 56
walk from: 16 to: 56
```

7.7 Srovnání

Výsledky jsou srovnány s daty publikovanými v článku [7].

Tabulka 7.11: Velikosti tříchpohybu

třída pohybu	v článku	vygenerované
Basketbal	14	-
Tanec	31	21
Skok	18	22
Běh	44	13
Sezení	14	-
Chůze	75	41
Kopnutí	-	24
Zvednutí míče	-	14
Položení míče	-	20
Stání	-	4

Jelikož moje testovací data nejsou stejná s daty, která použili autoři článku, výsledky nejsou testu zcela stejné.

Kapitola 8

Závěr

Vytvořil jsem prototyp programu pro automatickou kategorizaci pohybu. Algoritmy tohoto programu budou součástí systému umožňujícího snadnější a efektivnější vyhledávání a třídění velkého množství pohybových dat v animačních studiích.

Program je schopný pracovat s daty uloženými ve formátech FBX a BVH. Vytváří vlastní struktury a soubory naučených dat a popisů scén. Pro logická data využívá standardní formát XML a pro větší objemy dat pohybů využívá vlastní formát souborů.

Program obsahuje veškerou funkcionalitu potřebnou pro načtení pohybových dat. Uživatel má možnost přesně definovat struktury scén. Velkou výhodou je možnost modifikovat uložené naučené pohyby. Uživatel může ovlivnit chápání uložených dat a míru důležitosti jednotlivých prvků pohybu.

Funkce jsem testoval na osmi typech pohybů. Velikosti tříd pohybu, které algoritmus vytvořil v učící fázi, odpovídají mému předpokladu. Algoritmus dokáže správně rozpoznat důležité části scén a umí odfiltrovat pohyb méně významných prvků. Tento test zároveň odhalil určitý nedostatek implementace. Pokud jsou například při chůzi pohyby rukou stejné, algoritmus rozhodne, že jsou důležité pro určení třídy pohybu. Algoritmus z tohoto důvodu není schopen kategorizovat složené pohyby. Složeným pohybem mám na mysli např. pohyb, při kterém osoba jde a zároveň mává rukou.

Výsledný program je i přes nalezenou chybu schopný plnit základní požadavky. Problematika automatické kategorizace pohybu je velmi široká. Aby mohl být algoritmus plnohodnotně použitelný, je třeba tuto problematiku zkoumat a zpracovávat dlouhodobě.

V další fázi vývoje toho algoritmu je nutné opravit zjištěnou chybu a zvýšit přesnost porovnávání tříd s pohybem. Dalším úkolem je rozšířit schopnost algoritmu vybrat klíčové polohy, které jsou základem k nalezení nejdůležitějších prvků pohybu.

Seznam použitých zkratek

MOCAP	V angličtině „Motion Capture“.
FBX	Formát vyvynutý firmou Atodesk.
BVH	Formát ukládající kostru a pohybová data MOCAP.
3D	Označení trojrozměrného prostoru.
tag	Element v souboru XML.
Kinect	Elektronické zařízení využívající 2 kamery a hloubkový sensor prosnímání okolního prostředí.

Obsah přiloženého CD

```
CD
 |-- dokumentace                                  - Dokumantace diplomové práce
 |   |--latex                                     - Zdrojové soubory textu práce
 |   |--uml                                       - Diagramy projektu
 |   \--strnava1_motionclassification_2015.pdf    - Text práce v PDF
 |-- zdrojovy_kod                                 - Zdrojové kódy
 \-- MotionClassification.exe                     - Přeložená mobilní aplikace
 \-- readme.txt                                   - Informace o obsahu CD
```

Literatura

[1] *Struktura BVH kostry* [online]. Dostupné z: <http://lookslikematt.com/cgtalk/mb_bone-set-up_bvh_characterstudio.jpg>.

[2] *Stažení knihovny FBX SDK* [online]. Dostupné z: <http://usa.autodesk.com/adsk/servlet/pc/item?id=24314456&siteID=123112>.

[3] *Knihovna PugiXML* [online]. Dostupné z: <http://pugixml.org/>.

[4] BERKA, R. – TRáVNíčEK, Z. Extending MPEG-7 Description Schema to Annotate MOCAP Data. 2014.

[5] JAN, K. *Analytická geometrie* [online]. Dostupné z: <http://www.karlin.mff.cuni.cz/katedry/kdm/diplomky/jan_koncel/vektory.php?kapitola=nasobeniVektoruCislem>.

[6] LIU, F. et al. 3d motion retrieval with motion index tree.

[7] TIAN, Q. et al. A semantic feature for human motion retrieval. online in Wiley Online Library, 2013.